ЛЕКЦИЯ 2: Система типов. Чтение и вывод данных.

Стандартные типы данных: целочисленные

- синтаксически встроенные в язык (явл. ключевыми словами):

○ int - целые числа со знаком в диапазоне -32767...32767 (16 бит) [по стандарту диапазон не хуже, по факту - 32 бит]

Полный диапазон в дополнении до двух:

-32768...32767 - не явл. стандартом в Си, т.к. не помещается в 2 байта.

Битность процессора - размерность указателя (сколько данных можно адресовать) (сколько данных в адресе)

Процессор	тип int	
8 бит	16 бит	
16 бит	16 бит	
32 бит	32 бит	не стали изменять, т.к. много поломок
64 бит	32 бит	

- short (int) ≤ int (16 бит по факту)

при исп. модификатора можно не писать int

- long (int) - зависит от системы:

Windows - 32 бит

Linux → компиляция в 32-битный код - 32 бит
→ компиляция в 64-битный код - 64 бит

лучше не использовать без конкретной цели

- long long (int) ≥ 64 бит (64 бита)

Добавлен в C99. (крупные изменения в языке происходили в стандартах C99, C11, C23).

unsigned - модификатор: (тип такого же размера, что и тип, кодирующий число без знака)

unsigned int ⇔ int unsigned

Критично для маленьких диапазонов.

Применим ко всем рассмотренным типам.

НЕ исп. __int128

unsigned, short, long - модификаторы

- char (8 бит = 1 байт)

По стандарту ∄ байтов => измерение в char.

Unsigned / signed применимо к char.
(-128 ... 127) (0 ... 255)

Байт - минимальная адресуемая память (мин. кусок ОП).

Может быть как знаковым, так и беззнаковым

Если работаем с char, то всегда используем модификатор знака: unsigned / signed

Можно переключить знаковость char с помощью ключа компилятора.

- boolean (1 - true, 0 - false) (8 бит)

_Bool - хранит 2 значения; (8 бит, т.к. меньше не адресуется)
явл. беззнаковым

Си: регистрозависимый язык, так что int ≠ Int при объявлении типа данных.

Си: слова с _ и заглавной буквы - зарезервированные слова для потенциального расширения языка

Си: слова с __ - зарезервированные слова для компиляторных (компиляторозависимых) расширений.

Си: иногда в Си получается int, в то время как в Си++ - boolean.

bool - макрос при #include <stdbool.h>, не явл. ключевым (зарезервированным) словом.

true / false <stdbool.h>

→ рекомендованное использование bool

- доступные в подключаемых библиотеках:

<stddef.h>

← лучше для измерения размерностей

- size_t - для хранения размерности объектов в памяти. (<stddef.h>). Явл. беззнаковым
для индексации по масиву в памяти (если масив мб большой)
- ptrdiff_t - знаковый аналог size_t. (<stddef.h>)

Являются 32-ух битными / 64-ёх битными в зависимости от того, как компилир. программа.

для положения в файле - long long

для объектов в ОП: size_t, ptrdiff_t, int
без знака со знаком

<stdint.h>

○ intX_t , где t: 8, 16, 32, 64

В стандарте не гарантируется ∃ таких типов, т.к, например: если байт = 32 бит ⇏ int8_t

○ uintX_t, где t: 8, 16, 32, 64

intX_t и uintX_t рекомендованы к использованию!

○ _BitInt(X), где X - лимит компилятора

Применим модификатор unsigned ,
Произвольная битность: от 1 бит до стольки,
сколько поддерживает компилятор

Си: _BitInt(X) используется с C23 и только в Си.

Вывод типов фиксированной размерности.

```
x :          // целочисленные константы без объявления
                типа относятся к „ближайшему" типу,
x = 3u ;        в который помещаются.
```

модификатор unsigned (суфикс)

x = 3ul / или 3llu

суффикс long long

Суффиксы могут комбинироваться в произвольном порядке

x = 3-wb или 3u'wb ← знак единства

↑ может быть отдельно

x :

```
printf ("%  i ", x)
```

$\overset{\text{d}}{\text{u}}$ → только для printf!

спецификатор типа

% i - для знаковых

% u - для беззнаковых

На маленьких + значениях i = u,
на больших - i ≠ u.

x :

```
printf ("%  x ", x)
            X
            o - вывод в восьмеричной системе
```

?

x, X - для 16-ричной системы:

↓ ↓

1f 1F

Если тип не int с точки зрения числа:

- short, char

x:

```
             short
              ~     i
printf(" % h h      d   " , x ); | типы меньше int → int
           ~~~      u          для printf.
           char    x X
```

- long, long long

x:

```
            long
             ~      i
printf(" % l l      d  ", x );
                    u
           ~~~     x,X
        long long
```

- size_t, ptrdiff_t)

x:

```
              ↙ size_t
printf(" % z    i   " , x );
                d
           t    u
                x,X
              ↖ ptrdiff_t
```

модификатор длины

Вывод типов фиксированной размерности с „шашей"

```
# include <inttypes.h>
```

uint32_t x: вывод длинной константной строки

```
→ printf(" % " PRIu32 " abc\n", x );
                „u"
```

Си: пробел между % и PRI при отсутствии не даст ошибку компиляции в Си.

uint32_t x:

```
→ printf(" % u abc\n" );
```

Чтение типов данных фиксированной размерности.

uint32_t x:

```
scanf(" %      ", & x )
```

Чтение типов

int x:

```
             X, x ← шестнадцатиричная константа
scanf(" %    i"  ← распознаёт и то, и то
             d ← десятичная константа
             o ← восьмиричная константа
```

```
// x = 20   <=>  x = 0x14
10-ричная    16-ричная

             x = 0x1aB ; (<=> x = 0x1Ab)
```

ведущий ноль представляет число в восьмиричную константу.

Префиксы констант: „шаблончики":)

0b - двоичное число (i) с вниманием на 8-ричн. числа

0x - шестнадцатиричное число (x, X, i)

0... - восьмиричное число (i)

Стандартные типы данных: дробные

- float - single pr. (IEEE-754) (32 бит)

  $10^{\pm 38}$ степени - диапазон принимаемых значений.

  При маленьких значениях (< 10^9) int = float = int, при больших значениях неточности

- double - double precision (IEEE-754) (64 бит)

  $10^{\pm 308}$ - диапазон принимаемых значений

  int = double = int - полная точность. (53 бит)
  long = double ≠ double - неполная точность.

- long double → double precision (64 бит)
  или
  → quad precision (128 бит)
  или
  → extended precision (80 бит) может занимать или 96 бит, или 128 бит (степени 2)
  → точнее double
  → быстрее quad
  → медленнее double

  * не исп. в курсе

  $10^{\pm 4932}$ - диапазон принимаемых значений.

  half - стандартный тип (16 бит)
  bfloat - нестандарт. тип (16 бит)
  (старший кусок single precision)

Объявление констант:

x = 1,5 ; - double

x = 1,5 f ; - float

x = 1.5 l ; - long double

Чем меньше точность, тем больше операций ⇒ ⇒ быстрее работа программы.

_Float16 :

_Float32 :

Чтение и запись чисел с плавающей точкой:

```
„%  f  - float "
    lf - double
    Lf - long double
```

Си: есть поддержка комплексной арифметики.
(в C++ поддержка библиотек с КА)

Комплексная арифметика:

```
_Complex float x; ⇔ float _Complex x;
```

$$x = \underbrace{1.5f}_{\text{float}} - \underbrace{2.3if}_{\text{мнимая часть}};$$

Очень точная, так что работает долго.
Формула для нахождения комплексного числа: $\sqrt{re^2 + im^2}$ (есть проблема переполнения).

При выходе за пределы диапазона получим бесконечность (переполнение).

Компилятор Microsoft не поддерживает КА.

Унарный минус

```
int x = -2; // константа „2" с унарным минусом
    x = -128; // long,
    x = (-127 - 1) // int
```

Операция унарный минус определена и для signed, и для unsigned.

```
x = -2u; // некое + число, эквивалентное по
               модулю числа -2.
```

Создание unsigned int без страданий:

```
int x, y;

typedef unsigned int uint; // создание типа
   uint x; // создание x типа uint.
        unsigned int uint; // создание переменной
                              типа unsigned int
```

Автор:
Алимова Айгина,
М3132.
Вопросы, ошибки и предложения нужно писать сюда;
tg: @entelechyy